Notes from SSDS/ISI Interface and Metadata Design Meeting(s)
March 14th & 15th, 2002

Attendees:

- Dan Davis
- Duane Edgington
- Kevin Gomes
- John Graybeal
- Tom O'Reilly
- Keith Raybould
- Rich Schramm

SUMMARY

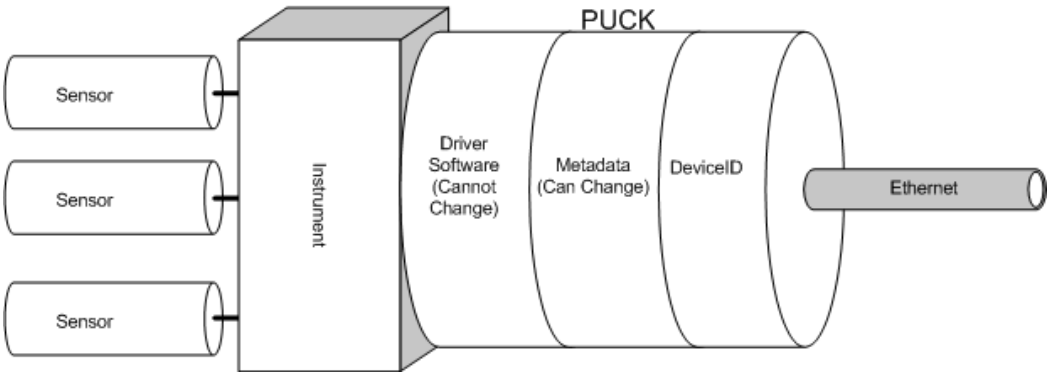
As a summary, the following lists are used to track the issues in these meetings. They are broken down into three categories, 'consensus', 'no-consensus', and 'to be discussed'. The 'consensus' list contains items that have been agreed to and understood by all members. The 'no-consensus' list contains issues that were discussed but are still outstanding and have not been agreed to by all members. The 'to be discussed' list contains all questions not yet discussed.

CONSENSUS:

1. **There is the possibility that software components (i.e. "soft-instruments") will be sending data to SSDS as well as ISI devices.**
2. **The ISI portal will manage bandwidth.**
3. **We do not need to build a system that is completely 'self-healing'. Essentially this means we do not have to account for every possible failure of software components in the system.**
4. **There is a qualification step (TBD) for ISI devices that have to be successful before they can be deployed on MOOS.**
5. **SSDS can define core SensorDataPacket and SensorStatusPacket architectural elements to meet their needs.**
6. **Whatever definitions result for SensorXxxPackets must support expected functional requirements for wet side clients of ISI.**

NO-CONSENSUS:

1. What does the logical model of a device look like? See the following figure:



2. What are the complete needs for data and metadata to define a single data packet and its relationships with other data (its interpretation)? This can be broken into the following questions:
    1. What is the metadata needed to define a data bucket?
    2. What does the state (or status) of a device entail and what data is needed to describe a state?
    3. Where are the possible sources of data and metadata that relate to a data packet?
    4. What portion of the total data/metadata needs is ISI responsible for providing?
    5. What is meant by a change of state (change in data, change in metadata, both)?
    6. Where can the data that defines the state of a device be located?
    7. Will there be data that is manually entered to complete the necessary data (for example if a lattitude and longitude cannot be entered on the puck)?
    8. Where will that manual data be stored?
    9. What information should be stored on the puck?
    10. Is the puck responsible for reporting it's "complete state" in the SensorStatusPacket?
    11. What data/metadata could/should be in a SensorStatusPacket?
    12. Could the SensorStatusPacket contain XML that defines the structure of the data within the SensorDataPackets and then the SensorDataPackets contain data in compact (non-XML) format?
    13. What data/metadata could/should be in a SensorDataPacket?
    14. Do we need a separate packet for any other information, or should we assume everything we need to know will be wrapped up in the XML that comes in the ISI packet container?
3. What is the data storage responsibility of the ISI portal?
4. If the portal is the data storage for all packets and SSDS accesses all packets from the portal, what is the exact guarantee of packet availability?
5. Is the packet availability guarantee at the portal or on the devices?
6. What is the list of responsibilities for the ISI portal?
7. Who is developing the ISI portal?
8. What is the interaction (interface) between the ISI portal and SSDS (define their respective responsibilities)?
9. If the ISI portal is developed and stores data at the portal, what additional functionality does SSDS provide?
10. Can clients interact directly with devices, or are all requests made to the portal?
11. Does the portal hand off device proxies to clients (appears possible from NMC UML diagrams) and are any of these interactions recorded in SSDS?
12. Does SSDS get all of its data from the portal, or does it need to get all the data from the devices and handle all device connections?
13. Is TCP/IP used for distributed object network?
14. Can we get agreement on a proposed definition: "Save all data = Capture all data that is presented by the device driver developer in the sensor packet stream"?. This question arose from the fact that there appears to be no way to collect and store non-core datastream data. For example, do we need to store exceptions, interactions between clients and devices (command and control), etc.
15. If we do need to store more than just the core data streams, how can we do that with the current ISI architecture? Can we use links (or containers) in the SensorStatusPackets describing the non-core data?
16. What levels of changes cause a deviceID to change (if any)?
17. Will metadata timestamp be enough, or do we need to revisit metadataRevision?
18. Will SSDS need to track a revision change history for the ISI deviceID (a sort of rolling history)?
19. If configuration changes how do we restore state after reset or power-ups?

20. Can anyone develop class extensions to the SensorPacket class and if so, what are the minimum requirements/policies?
21. How do we handle deviceIDs for "soft-instruments"?
22. We need to synchronize the definitions for the Metatdata ID between IAG and ISI
23. Do the soft-instruments also need to go through the ISI qualification step before they can send data to SSDS?

TO BE DISCUSSED:

Critical Metadata issues:

1. What is the data structure for child/parent hierarchical relationships? (Question #4 from the IAG Use Case "Update Configuration Metadata") -
2. Is SSDS responsible for tracking node and device hierarchy? (Question #3 from the IAG Use Case "Update Configuration Metadata") -
3. Who reports instrument state change, instrument (through the parent) or the parent itself? (Question #17 from IAG Use Cases) -
4. Do ISI nodes have device IDs? (Question #2 from IAG Use Case "Add ISI Node to MOOS") -
5. What is the metadata that is necessary to distinctly identify an ISI node? (Question #3 from IAG Use Case "Add ISI Node to MOOS") -
6. What metadata needs to be included in an event notification (is it any different from data or state metadata)? (Question #2 from IAG Use Case "Send Event Notification") -
7. Will ISI (SINs) track all the instruments software/firmware revisions, is SSDS supposed to, or will the revisions only be stored in the pucks? (Question #8 from IAG Use Cases) -

Important Metadata issues:

1. Is MBARI responsible for creating a specification that people can create pucks for use in MOOS against (if so, we have to provide a pretty solid metadata spec, and a good process for changing it). (Question from IAG Use Case "Obtain Puck") -
2. What information needs to be stored about nodes? (Question from IAG Use Case "Update Configuration Metadata") -
3. What information needs to be stored about devices? (Question from IAG Use Case "Update Configuration Metadata") -
4. If SSDS and other clients are to interact with devices directly, how will clients get the necessary Java classes to interact with devices (Stepping through the sequence of operations when configuration metadata is updated 'on the fly' (while the instrument is in place) is probably a valuable exercise)? (Question from IAG Use Case "Update Configuration Metadata") -
5. Will software be provided to device developers to allow them to communicate with the puck, or is this something they will have to develop (need to have common understanding of how/when this data is changed)? (Question #2 from IAG Use Case "Update Device Parameters") -
6. Does ISI transmit the metadata to shore when the device is installed? (Question from IAG Use Case "Access Device Metadata") -
7. Does ISI transmit an event to SSDS, then SSDS would need to explicitly request the device metadata on the evens arrival (we need to understand the sequence of events here and the specific way MOOS information arrives at SSDS (is it sometimes via method calls, or always via data arrival?). A sequence diagram with actual classes might be helpful.)? (Question from IAG Use Case "Access Device Metadata") -
8. How would a 'log an event' work (this event should include the details of the device addition. I.e. send an event notification that a new device was installed to a shore-side ISI Component. As above, need to understand the sequence)? (Question from IAG Use Case "Install Device")
9. Does each individual device keep track of its subscribers, or is the event sent all the way up to the SIN and the SIN sends out the event notifications to the subscribers (another question about how data moves through the system; this one is more architectural than metadata, but would also be helpful to understand the sequence)? (Question from IAG Use Case "Send Event Notification") -
10. In response to some add device events, certain information may need to be entered into the system manually. Items such as final deployment nominal depth, lat/lon, access permission, etc., may need to be added by the operations technician. Should this be done via ISI events or SSDS manual interface (we should make sure reasonable processes exist for creating/logging this metadata)? (Question from IAG Use Case "Detect Device Addition") -

11. Who is reponsible for managing and generating device ID's, ISI or SSDS (or other)? (Question from IAG Use Case "Get New DeviceID) -
12. How are we going to guarantee the device ID is unique? (Question from IAG Use Case "Get New DeviceID) -
13. How do we assign device ID's to software components (non-ISI instruments)? (Question from IAG Use Case "Get New DeviceID) -
14. What physical changes require a new device ID to be generated? <Question to address by process. Who should develop?> (Question #1 from IAG Use Case "Physically Modify Device") -
15. Does SSDS need to store all the information about exceptions that were thrown in the distributed network of nodes and devices (Raises more general question about logging by software, mechanisms to do that and to ID the software)? (Question #10 from IAG Use Cases) -
16. How will we uniquely associate data with metadata (this is really a question aimed at SSDS, if I'm not mistaken)? (Question #1 from IAG Use Case "Receive Data") -
17. Does/can ISI generate messages when the heartbeat to instruments is lost (It isn't critical to SSDS that this information be available, but it might be the kind of operational information that would be very useful for scientists and operators to have ... oops, but not a metadata question)? (Question from IAG Use Case "Remove ISI Node from MOOS") -
18. Logged events include a timestamp of the time of the event. Notification of events are passed to the on-shore ISI node which, in turn, passes it on to SSDS? (Question from IAG Use Case "Log Event") -

Definitions (need to be reviewed and agreed to):

1. instrument ID: a unique identifier assigned to each instantiation of an instrument ; never duplicated within SSDS and never changes over the life of that instrument and over the life of SSDS
2. process ID: (analogous to an instrument ID) a unique identifier assigned to each version of data-generating software; never duplicated within SSDS and never changes across various instantiations of the same executable; must include enough information to identify the configuration-controlled source from which the process was created
3. personal ID: a unique identifier of the person providing data to the SSDS (for example, the Research Technician who performs a calibration process and enters the calibration data)
4. data source ID: a unique identifier defining the source of data coming to the SSDS; typically either an instrument ID, process ID, or personal ID
5. metadata ID: a unique identifier assigned to a record content description which applies to a specific class of SSDS data record or SSDS data packet (different versions of record content descriptions may have the same metadata ID, if they all apply to the same class of data records/data packets; in this case they will need metadata ID version numbers

March 14th Notes:

Keith started the meeting by having the group clearly define the focus and purpose of the meeting. Duane was appointed to lead the meeting and he started by clarifying that the main focus should be the metadata needs of SSDS and ISI. The group agreed and also agreed that the use case document and associated questions generated by the SSDS team for the meeting were not focused solely on the metadata, but on some general system questions as well. Since the ISI group brought a sheet with responses to some of the general questions in the use cases document, it was decided that the group would stay focused on the metadata.

The focus of the meeting was then trying to decide what the structure of the metadata should be in order to meet all the needs of the SSDS system and once that is defined, can the ISI system, in its current form, meet those needs. Essentially the focus of the discussion was to try and ensure that there was a tractable process for SSDS to rebuild the data stream coming from the devices on the ISI side of MOOS. The discussion focused on "given any data packet, how can we directly tie that packet to all the data and metadata which affect its interpretation?". After discussing some general issues, Tom went through a description of the ISI system and its operation. The following diagram was generated from that discussion:

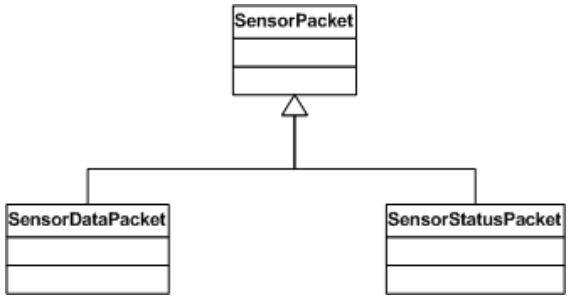


Figure 1 - ISI Sensor Packet Classes

The classes are as follows:

- SensorPacket - This class represent a generic packet that is being sent from an ISI device. It contains methods and attributes that are common among all packets generated by the ISI device.
- SensorDataPacket - This packet contains the data that is measured by the ISI device (i.e. temperature, pressure, etc.)
- SensorStatusPacket - This packet contains some data that identifies a portion of the state of the ISI device.

The packets will travel in a stream from the device and, as an example, will look something like this:

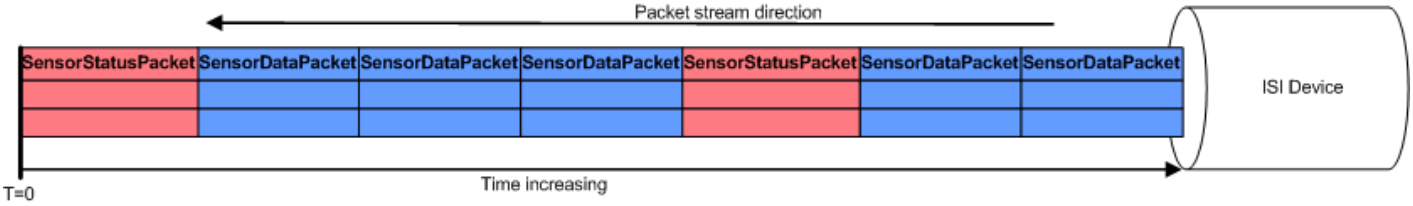


Figure 2 - Datastream Structure

The normal datastream will consist of SensorDataPackets that contain the data that the device is measuring. When something occurs that changes the state of the device, a SensorStatusPacket is generated and sent. As the discussion continued, several issues arose that pointed to a question about needing to track every piece of of electronic data that passes over the distributed network of devices. In other words, does SSDS need to store everything that happens on the ISI network of devices, or are we just concerned with capturing the "core (scheduled?)" datastreams that are being delivered by the devices? This was a question that, following further refinement, needed to be addressed by the science community, the MOOS steering committee, or both.

The questions then focused on the metadata needed to satisfy certain constraints. One constraint was that we needed to be able to guarantee the chronological order of the packets. It was decided that a timestamp alone was not enough to determine if the packets were in the correct order. A sequenceNumber (already a part of the ISI design), would be needed to ensure that the packets were in the correct order. This sequenceNumber would be a incrementing number that would be included at the SensorPacket level so all the packets generated had a sequenceNumber. The next issue was to decide how SSDS would know when packets are missing from the stream or request. Again, the sequenceNumber, which increments by one for each packet generated, could tell the system when there were missing packets. This would be done by looking over the sequenceNumber's and when there were gaps in the numbers, packets would be missing. In addition, a timestamp will be an attribute of each packet to serve as a redundant source to provide order to the sensor packets. In order to uniquely tie a packet to a device, the deviceID is provided with each packet. The deviceID is tied to a specific device and can uniquely identify the device the packet came from.

The revised look at the classes shows the following:

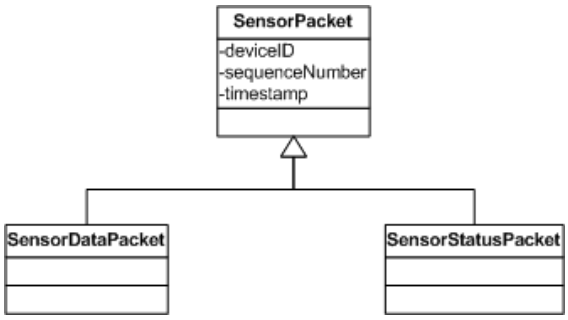


Figure 3 - Packet Classes with Attributes

At this point the question still remained about being able to 'directly' tie data packets back to the associated metadata packet. Essentially what was desired is for the packets to have the data included that would point directly to a metadata packet without having to perform any searching or inter/extrapolating. A proposed "metadata revision number" was examined and would become a part of the SensorPacket class (so that the SensorStatusPackets would include their own revision numbers). The class structure would then look like the following:

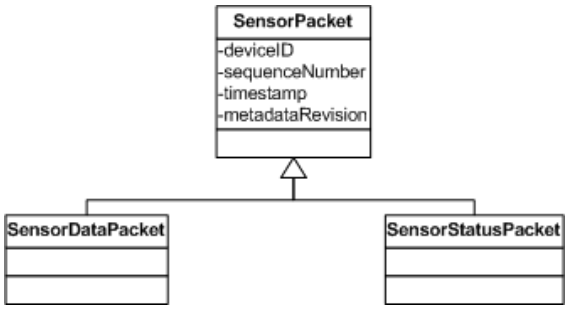


Figure 4 - Packet Classes with Metadata Revision Number

This was the breaking point from the first meeting and after the meeting, the IAG members distilled the pertinent questions from the use cases and brought those to the next meeting.

March 15th Notes:

The next meeting started with the list of questions from IAG for review and recap.

1. *How are packets going to be arriving to SSDS? (Question #12 from IAG Use Cases) -*
   All MOOS packets will arrive at the ISI portal and the portal brokers the requests for these data packets. All clients (including SSDS) will go to the portal to obtain the data packets and make requests. The data portal is a shore based system and the clients will only need to go to one place to get the data. This appeared to have redundancy with SSDS functionality so question was never resolved.

2. *Will the packets be in chronological order? (Question #12a from IAG Use Cases) -*
   Usually, but not guaranteed. All the packets will eventually be available in chronological order at the portal, but this is only guaranteed within a reasonable time frame. Again, this was not finalized due to the confusion between the data storage requirements of the portal verses the SSDS system.

3. *Is there an arrival guarantee on the packets? (Question #12b from IAG Use Cases) -*
   Yes, this is a requirement of the ISI system. The actual guarantee was never defined, is it at the portal or at the devices?

4. *If a gap in the sequence is detected and there is suspicion of dropped packets, can SSDS requery the the instrument for lost data packets? (Question #12c from IAG Use Cases) -*
   Yes, the client (SSDS in this case) would request packets from a certain deviceID by time range. Currently there is no way to ask for a packet by sequenceNumber, but that could be added

later.

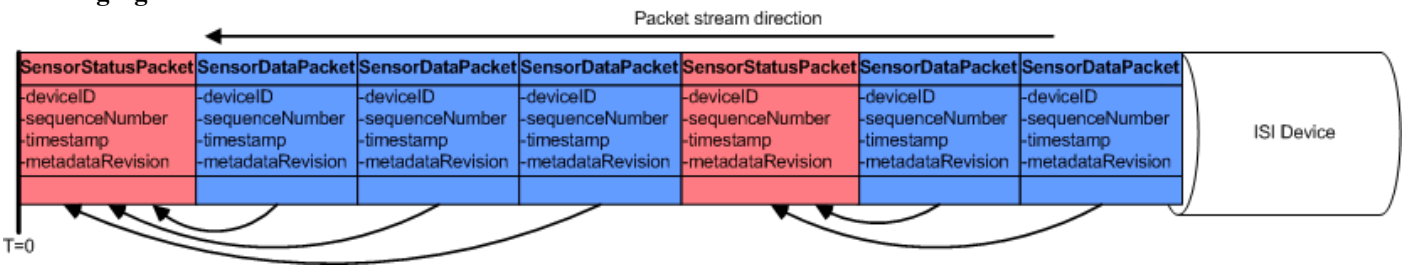
5. *How will we know when packets get dropped? (Question #12d from IAG Use Cases) -*
   By checking the sequenceNumber. The sequence numbers should be complete (each number increments by one) and if there is a gap in the sequence number, packets are missing.

6. *Can we/are we using TCP/IP? (Question #12e from IAG Use Cases) -*
   This was never finalized. The discussion was that maybe TCP/IP is going to be used, but not sure.

7. *Can we request specific (individual) packets? (Question #12f from IAG Use Cases) -*
   Yes, but only by time range. There is currently no way to request by sequenceNumber/deviceID/timestamp to get a specific packet, but this could be added. Again, this request is made to the portal, not the device itself.

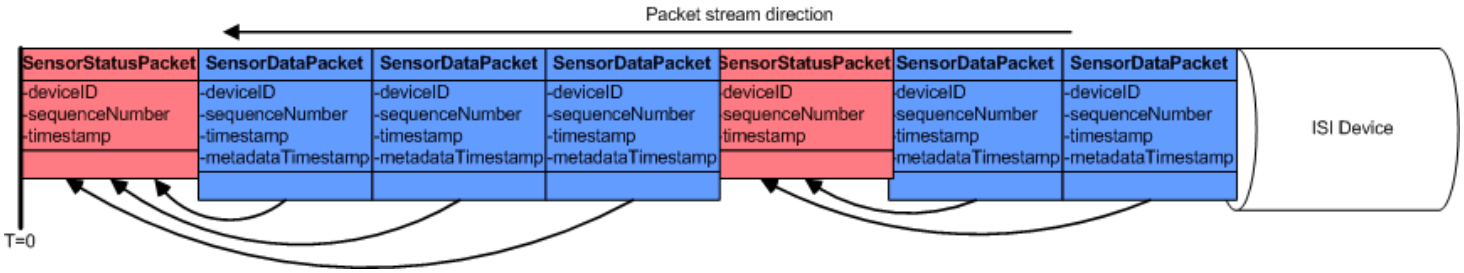
8. *What is the requirement for data storage/access on the "wet side" and how will this be managed? (Question #12g from IAG Use Cases) -*
   The portal has not been defined to this point, so this is an unknown.

9. *Can we uniquely tie every packet back to it's metadata and if so how? (Question #13 from IAG Use Cases) -*
   This can be done with the combination of deviceID and metadataRevision. This is shown in the following figure:



10. *What uniquely identifies data and metadata packets?*
    The combination of deviceID, sequenceNumber, metadataRevision, and (probably) timestamp is a start. This topic needs further elaboration.

11. *What is the process for getting configuration data into the puck? (Question #20 from IAG Use Cases) -*
    This question in conjunction with 9 and 10, created a great deal of discussion and follow up questions. The stem of the discussion was the issue of configuration data and its persistence on the puck. The current ISI implementation does not keep configuration changes over re-starts of the device. In other words, there is some initial state that the device is programmed in a "read-only" memory. When a device is re-started, it returns to its initial state. In this form, it is not easy to implement a metadata revision number due to the lack of persistent storage. The question branched out into other questions about state vs. non-state parameters. Currently, the puck is not configurable in the field by design. This prevents changes that would make the device inoperable in the field. The idea is that any information that would require a change to the puck software itself, is major enough that the device should be removed, reprogrammed on shore, the deviceID changed and then deployed back in the field. Other small configuration changes could be made that would only cause the issuing of new SensorStatusPackets. This generated more questions which are listed below in the 'No-consensus' list of questions. Since a metadata revision number was not a part of the current implementation, a question was then posed about an attribute in the SensorDataPacket that would contain the timestamp of the last relevant SensorStatusPacket. That would give a direct link from the SensorDataPacket to the SensorStatusPacket. This changes the view of the figure from question 9 to the following:



The concern was then raised that this method would create unnecessary 'buckets' of data on the SSDS side. For example, if something changes in the metadata, normally that would cause a change in the way the data sets are organized in SSDS. But if SSDS creates these 'buckets' of data based on SensorStatusPackets, SSDS would be creating new buckets everytime a SensorStatusPacket is sent. However, the change made to cause the SensorStatusPacket to be sent may not be of the type to require a new data 'bucket' and would cause unnecessary fragmenting of the data. The idea of an external link (or container) to define what actually changed was discussed again. This would enable SSDS to look at why the SensorStatusPacket was sent and determine if a new data 'bucket' is necessary. A further refinement to that idea was suggested. Since the SensorStatusPackets are merely containers for data, maybe SSDS could look at the contents of the container and decide if the data has changed enough to warrant a new 'bucket'.

12. *Who defines the format and usage of the metadata in the puck? (Question #22 from IAG Use Cases) -*
MOOS should be responsible for developing the DTD/Schema for the puck metadata, but the instrument developer and their customer are responsible for what format of the metadata is issued by the device. The developers of the wet-side and shore-side software are responsible for the programmatic usage of the metadata. All users that want to use the data are responsible for the operation and archival usage of the metadata. This quickly led to another question about the metadata. What is contained in the metadata that can change and what is contained in the puck driver software?

13. *How do we guarantee the parameters stored in the puck and those stored in portal and SSDS are synchronized? (Question #1 from IAG Use case "Update device parameters") -*
The information in the puck is tested on a bench, and this is where the "image" of the driver on the puck should be captured. Once the puck is taken off the bench, it cannot change. If the puck changes, a new deviceID is issued. Again this generated more questions that are listed in the 'no-consensus' list below. This was only one proposal. Another option might be that the synchronization would be inevitable through normal operation of the system (in which metadata is supplied by the puck as part of the data stream, and the portal picks it up).